

ANUPLACE: A SYNTHESIS AWARE VLSI PLACER TO MINIMIZE TIMING CLOSURE

Santeppa Kambham¹ and Krishna Prasad K.S.R²

¹ANURAG, DRDO, Kanchanbagh, Hyderabad-500058, India

²ECE Dept, National Institute of Technology, Warangal-506004, India

ABSTRACT

In Deep Sub Micron (DSM) technologies, circuits fail to meet the timings estimated during synthesis after completion of the layout which is termed as 'Timing Closure' problem. This work focuses on the study of reasons for failure of timing closure for a given synthesis solution. It was found that this failure is due to non-adherence of synthesizer's assumptions during placement. A synthesis aware new placer called ANUPLACE was developed which adheres to assumptions made during synthesis. The new algorithms developed are illustrated with an example. ANUPLACE was applied to a set of standard placement benchmark circuits. There was an average improvement of 53.7% in the Half-Perimeter-Wire-Lengths (HPWL) with an average area penalty of 12.6% of the placed circuits when compared to the results obtained by the existing placement algorithms reported in the literature.

KEYWORDS: Placement, Signal flow, Synthesis, Timing

I. INTRODUCTION

VLSI IC design process involves two important steps namely (i) synthesis of high level representation of the circuit producing technology mapped components and net-list and (ii) layout of the technology mapped circuit. During the layout process, the placement of circuit components to the exact locations is carried out. The final layout should meet the timing and area requirements which are estimated during the synthesis process. Placement is the major step which decides the area and delay of the final layout. If the area and delay requirements are not met, the circuits are to be re-synthesized. This two step process has to be iterated till the required area and delay are achieved. In Deep Sub Micron (DSM) technologies, circuits fail to meet the timing requirements estimated during the synthesis after completing the layout. This is termed as "Timing Closure" problem. It has been found that even after several iterations, this two step process does not converge [1,2,3]. One reason for this non-convergence is that the synthesis and layout are posed as two independent problems and each one solved separately. There are other solutions which try to unify these two steps to achieve timing closure which can be classified into two categories (i) synthesis centric[4,5,6] and (ii) layout centric [7,8]. In synthesis centric methods, layout related information is used during synthesis process. In layout centric methods, the sub modules of circuits which are not meeting the requirements are re-synthesized. All these methods have not investigated why a given synthesis solution is not able to meet the timing requirements after placement. Our work focuses in finding the reasons for failure of timing closure for a given synthesis solution. Based on these findings, we developed a placer named as ANUPLACE which minimizes the timing closure problem by placing the circuits as per the assumptions made during the synthesis process.

In Section 2, we briefly review the existing methods of placement and their limitations. Section 3 tabulates and illustrates the reasons for failure of timing closure. Section 4 describes the implications in adhering to synthesis assumptions during placement. Based on this, the basis for new placement algorithm was worked out in Section 5. With this background, a new placer called ANUPLACE was developed which is described in Section 6. The new placer ANUPLACE is illustrated with an example in Section 7. Improvements to initial placement solution are given in Section 8. The experimental setup to evaluate ANUPLACE is described in Section 9. Results are tabulated and improvements obtained are discussed. Conclusions of research work carried and future scope are given in Section 10.

II. EXISTING PLACEMENT METHODS AND THEIR LIMITATIONS

Placement assigns exact locations to circuit components within chip area. The existing algorithms use component cell dimensions and component interconnection information as input to the placer. Thus the placer is not directly coupled to the synthesis. Lot of information available after synthesis is not used during placement [9,10,11,28,36,37]. The studies in [12] show that the results of leading placement tools from both industry and academia may be up to 50% to 150% away from optimal in total wire length.

Major classical approaches to placement are Constructive method and Iterative method [13]. In Constructive placement, once the components are placed, they will never be modified thereafter. The constructive methods are (i) Partitioning-based (ii) Quadratic assignment and (iii) Cluster growth. An iterative method repeatedly modifies a feasible placement by changing the positions of one or more core cells and evaluates the result. It produces better result at the expense of enormous amounts of computation time. Main iterative methods are (i) Simulated annealing, (ii) Simulated evolution and (iii) Force-directed. During placement, we have to optimize a specific objective function. Typical objectives include wire length, cut, routing congestion and performance. These classical approaches are very effective and efficient on small to medium scale designs. In DSM SOC era, due to complex chips and interconnect delay dominance, these are not very effective [1,2,3,4]. Some new methods to overcome this problem reported in literature [13] are (a) Hierarchical placement, which utilizes the structural properties [23] of the circuit during placement (b) Re-synthesis, which re-synthesizes a soft-macro, in case of timing violation. (3) Re-timing method relocates registers to reduce the cycle time while preserving the functionality. Existing timing-driven placement algorithms [14,15,16,17,18,19] are classified into two categories: path-based and net-based. Path-based algorithms try to directly minimize the longest path delay. Popular approaches in this category include mathematical programming and iterative critical path estimation. TimberWolf [18] used simulated annealing to minimize a set of pre-specified timing-critical paths. The drawback is that they usually require substantial computation resources. In the net-based algorithms, timing constraints are transformed into net-length constraints. The use of signal direction to guide the placement process found to give better results [28]. In Timing driven placement based on monotone cell ordering constraints [24], a new timing driven placement algorithm was presented, which attempts to minimize zigzags and criss-crosses on the timing-critical paths of a circuit.

Table 1 summarises how the existing algorithms are unable to solve the timing closure problem for a *given* synthesis solution. Most of the existing placement algorithms consider only connectivity information during placement and ignore other information available from synthesis [28].

III. REASONS FOR FAILURE OF TIMING CLOSURE

Our study has indicated that the failure of achieving timing closure is due to non-adherence of synthesizer's assumptions during placement. The assumptions made during synthesis [25,26,27,29] and the implications of these assumptions during placement are summarized in Table 2 and illustrated in Figures 1 to 8. Column 1 with heading "Fig" refers to the Figure number.

Table 1 Drawbacks of existing placement methods

Placement method	Drawback
Quadratic assignment [20]	Minimizes all wires whereas only critical path is to be minimized
Cluster growth [21]	Loosing track of cells along signal flow
Simulated annealing [18]	Signal flow disturbed
Force directed [22]	Tries to minimize all wires which is not required
Hierarchical placement [23]	Global signal flow not known. Additional burden of partitioning into cones
Re-synthesis of soft-macros [8]	Iterative process
Monotone cell ordering [24]	Additional burden of finding zigzags and criss-crosses from net-list

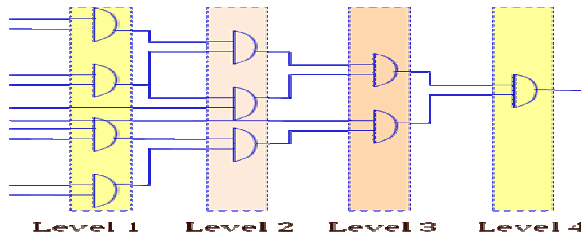


Figure 1 Gates placed as per levels

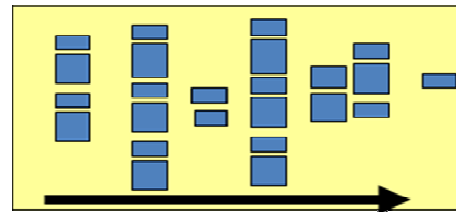


Figure 2 Non-uniformity of row widths

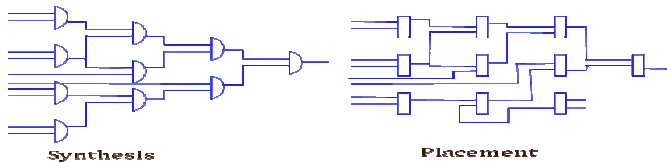


Figure 3 Cones versus Rectangle

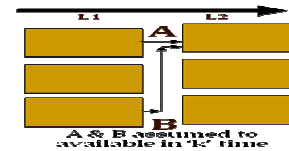


Figure 4 Primary inputs

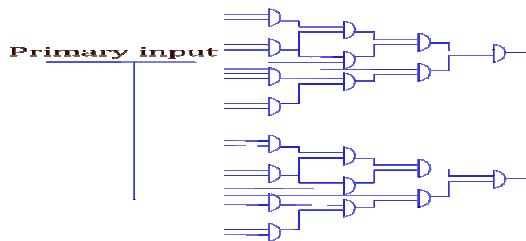


Figure 5 Sharing inputs

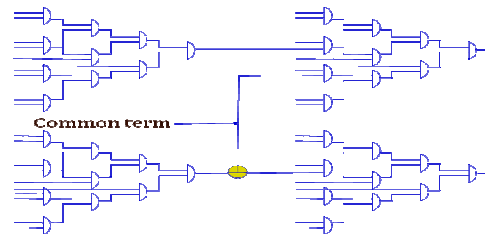


Figure 6 Sharing common terms

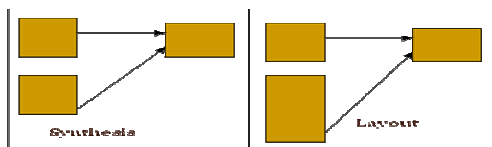


Figure 7 Non-uniformity of cell sizes

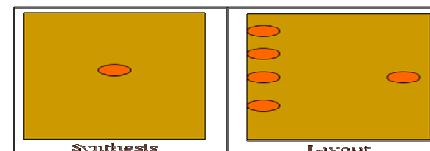


Figure 8 Pin positions on cell

Table 2 Implication of non-adherence of synthesis assumptions

Fig	Synthesis Assumption	Placement Implication
1	Gates are placed as per levels.	During placement gates are randomly placed. This increases the delay in an unpredictable manner.

1	Delay is proportional to number of levels.	Since gates are randomly placed, delay is no longer proportional to number of levels.
1	Delay from one level to the other level is a fixed constant (some "k")	Since the original structure of levels is not maintained, the delay from one level to the other level is unpredictable.
1	Upper bound of delay = No. of levels * (Delay of max (level) + delay from one level to the next level)	Since the original structure of levels is not maintained, upper bound of delay is not predictable.
2	No restrictions on the aspect ratio- number of rows and columns	Synthesis assumes irregular structure as shown in figure. Placer tries to achieve rectangular shape. Due to this, synthesis assumptions here can never be met, if the goal is a rectangle.
2	No restrictions on the aspect ratio, no uniformity on size of rows or columns	Synthesizer has no notion of shape of the placement. It does not bother about uniformity on the size of rows or columns. Thus synthesizer may produce irregular shapes when it calculates delay. This is not the case with placer.
3	Synthesizer assumes a 'cone'.	Combinational circuits have a natural 'cone' shape as shown in figure. Placer requires 'rectangle' for effective use of silicon. Synthesizer expected delay can be achieved only if placer uses 'cone' for critical signals.
4	Geographical distance of input source pins	In the Figure, A & B assumed to be available in a constant 'k' time. In reality, this can never be the case. This synthesis assumption can never be met.
5	Sharing of inputs	Synthesizer assumes inputs to be available in constant time which is not the case during placement. This synthesis assumption can never be met.
6	Common terms	Sharing output produces more wire during layout than what was assumed during synthesis. This synthesis assumption can never be met.
7	Non-uniformity of cell sizes	Requires more wire during placement. Cell size (length and width) are uniform and fixed during synthesis as far as wire required for routing are concerned. This synthesis assumption can never be met.
8	Pin position on a cell	It is assumed that inputs are available at the same point on the cell. This is not the case during placement. This synthesis assumption can never be met.

IV. IMPLICATIONS IN ADHERING TO SYNTHESIS ASSUMPTIONS DURING PLACEMENT

We now analyze how we can adhere to synthesis assumptions during placement. Synthesizer assumes that cells are placed as per the levels assumed during synthesis, whereas during placement cells are placed randomly without any regard to levels. Cells can be placed as per the levels as a 'cone' [28] and use the left over area to fill with non-critical cells to form rectangle for better silicon utilization. Synthesizer assumes that delay is proportional to number of levels, where this information is lost during placement due to random placement. By placing cells on critical paths, as per the levels along signal flow, we adhere to this synthesis assumption. Non-critical cells can be placed in the left-over area. By placing cell as per levels assumed during synthesis, the cell from one level to the next level can be approximately maintained as a fixed constant. The upper bound of delay can be predicted. Synthesizer assumes irregular structure as shown in Figure 2. Cells which are not in critical paths can be moved to other row to achieve rectangular shape. Based on the above analysis, the basis for the new method is evolved, which is explained in the next section.

V. BASIS FOR THE NEW ALGORITHMS

The new method is evolved based on the following.

- Use natural signal flow available during synthesis [28].
- Use cone placement for signals along critical path [28].
- Try to maintain the placement as close to the levels assumed during synthesis.

Signal flow indicates the direction of signal, from Primary Input (PI) to another gate input or from output of a gate to the input of another gate. The issues in placing cells along signal flow are explained below with the help of Figure 9. The gate G has one output and 3 inputs.

S1, S2, S3 show the direction of signals to the inputs of G. Ideally the output of preceding gates should be on the straight lines S1, S2, S3 as shown in Figure 9 for the gate g1, g2, g3. The gates g1, g2, g3 are to be placed as close as possible to G. The pin separations w1, w2 are much smaller than gate widths f1, f2, f3 for gate g1, g2, g3. It is impossible to place all input gates g1, g2, g3 in a row in Level i such that their outputs fall on the straight lines s1, s2, s3. At least two out of 3 gates are to be placed as shown in Figure 10. This results on two bends on signals s1 and s3. This cannot be avoided. There can be only one signal which can be placed on the straight line. This can be used for placing critical paths. Other less critical paths can be placed above or below of this straight line. The new placement algorithms which are used in ANUPLACE are explained in the next section.

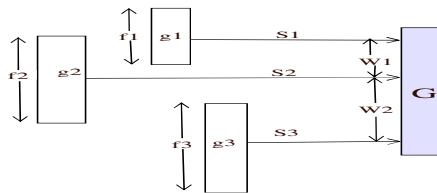


Figure 9 Signal Flow as per Synthesizer

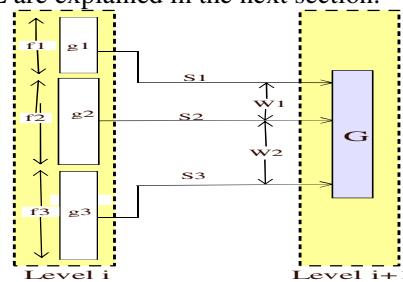


Figure 10 Placement along Signal Flow

VI. ALGORITHMS USED IN ANUPLACE

ANUPLACE reads the benchmark circuit which is in the form of a net-list, taken from “SIS” synthesizer [35], builds trees with primary outputs as roots as shown in Figure 11 and places the circuit along signal flow as cones. The placement benchmark circuits in bookshelf [30] format contain ‘nodes’ giving aspect ratio of gates and ‘nets’ which give interconnection details between gates and input/output terminals. These formats do not identify primary inputs or primary outputs. We took benchmark circuits from SIS [35] synthesizer in “BLIF” format which are then converted into Bookshelf format using converters provided in [31,32,33,34]. This produces “.nodes” and “.nets” file. The ‘nodes’ file identifies primary inputs and primary outputs by “_input” and “_output” suffix respectively. The “.nodes” file consists of information about gates, primary inputs and outputs. The “.nets” file consists of inter connections between the nodes and inputs/outputs. While parsing the files, Primary Input/Output information is obtained using the “terminal” names which identify “input/output”. The new placement algorithm is shown in Figure 12. Once the trees are created, delay information is read into the data structure, from SIS, which is used during placement. This delay information is available at every node from “SIS” synthesizer. A circuit example with 3 Primary outputs, marked as PO-1, PO-2 and PO-3 is shown in Figure 11.

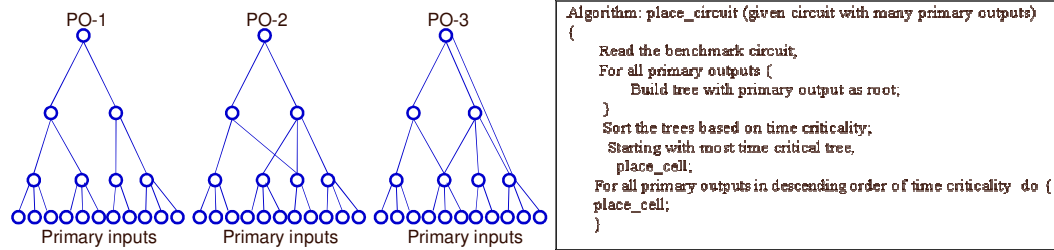


Figure 11 Trees with Primary output as root Figure 12 ANUPLACE algorithm

ANUPLACE works as follows.

- Read the benchmark circuit which is in the form of a Net-list with timing information.
- Build trees with primary outputs as roots.
- Sort the trees based on time criticality.
- Starting with most time critical tree, place on the layout surface, one tree pointed by the root starting from the primary output using “place-cell” algorithm shown in Figure 13.
- Place the remaining trees one by one, on the layout surface using “place-cell”.

The place_cell algorithm shown in Figure 13 works as follows.

- Place the cell pointed by root using “place_one_cell” algorithm shown in Figure 14.
- Sort the input trees based on time criticality;
- For each input, if it is a primary input, place it using “place_one_cell”, if not; call “place_cell” with this input recursively.

```

place_cell (tree pointed by root)
{
  place_one_cell (root);
  Sort the input trees based on time criticality;
  Starting with most critical input tree,
  For all inputs {
    If input is primary-input then
      place_one_cell (primary-input)
    Else place-cell (input)
  }
}

```

```

Algorithm: place_one_cell (cell)
{
  Check available place either on left or
  right or top or bottom to current location.
  Place cell at the available location along
  signal flow,
}

```

Figure 13 Algorithm Place-cell

Figure 14 Algorithm Place-one-cell

The “place_one_cell” algorithm shown in Figure 14 works as follows. The layout surface is divided into number of rows equal to number of levels in the tree as shown in Figure 11. Each row corresponds to one level of the tree. The first root cell is placed in the middle of the top row. Subsequently the children are placed below this row based on availability of the space. Roots of all trees (that is, all Primary Outputs) are placed in the top row. While placing a cell beneath a root, preference is given to the place along the signal flow. If space is not available on the signal flow path, then a cell is shifted either to right or left of the signal flow and placed as nearer as possible to the signal flow.

VII. ILLUSTRATION OF ANUPLACE WITH AN EXAMPLE

The ANUPLACE algorithms are illustrated with an example whose logic equations are shown in Figure 15. The timing information from the SIS synthesizer [35] is given in Table 3. The tree built by ANUPLACE with the average slacks is shown in Figure 16. The sequence of placement based on the time criticality is also shown in Figure 16. The sequence of placement is indicated by the numbers 1-

31 shown at the each node of the tree. The placement sequence number for each gate is also shown in the last column of Table 3. The initial placement is shown in Figure 17.

```

INORDER = a5 a6 a8
          a9 a10 a20;
OUTORDER = a1;
[127] = a20 + a5;
[15] = a10*a20;
[14] = [127]*[15];
[18] = a9*a10;
[19] = a5*a10;
[17] = [18]*[19];
[143] = [17] + [14];
[4] = a5*a6;
[82] = a9 + a8;
[135] = [02] + [4];
[119] = a10 + a6;
[12] = a5*a9;
[30] = [119]*[12];
[2] = [135]*[30];
[a1] = [143]*[2];

```

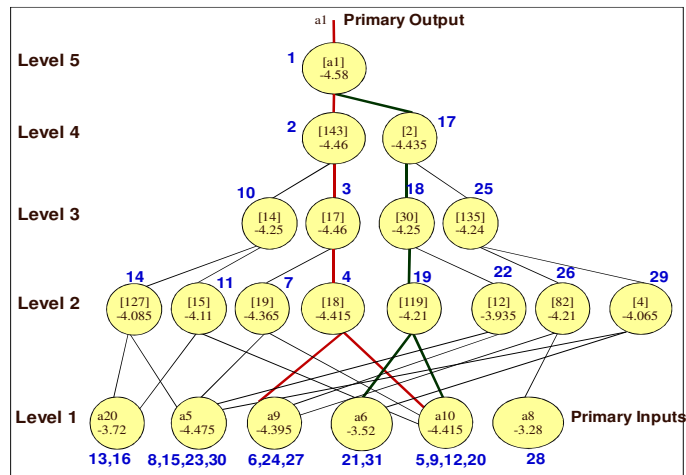


Figure 15 Example-Equations

Figure 16 Example-Tree built by ANUPLACE

There are 6 primary inputs marked as a5 a6 a8 a9 a10 a20 and there is one primary output marked as a1. There are 15 two input gates marked as [127], [15], [14], [18], [19], [17], [143], [4], [82], [135], [119], [12], [30], [2] and [a1]. The interconnections are as shown in Figure 16. The slack delays computed by the synthesizer at each gate are shown in Figure 16. The placement algorithm given in Figure 12, places the Primary Output cell a1 first. Then it looks at its leaf cells [143] and [2]. From the time criticality given in Figure 16, it places cell [143] along the signal flow just below the cell [a1]. Then the algorithm is recursively invoked to place the tree with root as [143] which places the cells and the inputs in the sequence [17], [18], a10, a9, [19], a5, a10, [14], [15], a10, a20, [127], a5 and a20 along the signal flow as shown. Once the placer completes the placement of tree pointed [143] as root, it starts placing the tree pointed by cell [2]. Now the cells marked [2], [30], [119], a10, a6, [12], a5, a9, [135], [82], a9, a8, [4], a5 and a6 are placed. This completes the placement of complete circuit. Primary Inputs and Primary Outputs are re-adjusted after placing all the cells.

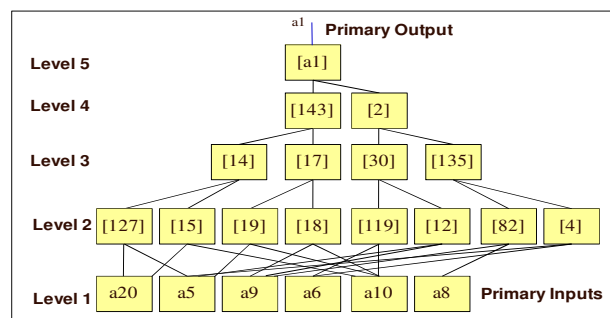


Figure 17 Example Initial placement

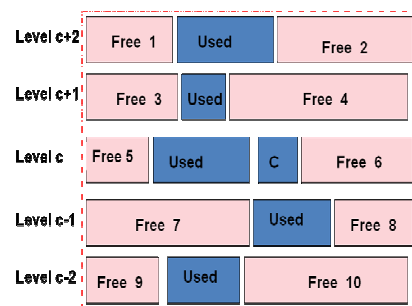


Figure 18 Find-best-place

Table 3 Timing information for the example circuit

Gate	Arrival time rise	Arrival time fall	Required time rise	Required time fall	Slack rise	Slack fall	Slack average	Placement sequence
a5	1.45	1.11	-3.28	-3.11	-4.73	-4.22	-4.475	8,15,23,30
a6	0.69	0.53	-2.89	-2.93	-3.58	-3.46	-3.52	21,31
a8	0.35	0.27	-2.84	-3.1	-3.19	-3.37	-3.28	28

a9	1.16	0.89	-3.47	-3.27	-4.63	-4.16	-4.395	6,24,27
a10	1.5	1.15	-3.44	-2.74	-4.94	-3.89	-4.415	5,9,12,20
a20	0.8	0.61	-3.51	-2.52	-4.31	-3.13	-3.72	13,16
[127]	1.72	2.18	-1.74	-2.53	-3.46	-4.71	-4.085	14
[15]	2.08	2.09	-1.71	-2.35	-3.79	-4.43	-4.11	11
[14]	3.13	2.64	-1.58	-1.14	-4.71	-3.79	-4.25	10
[18]	1.93	2.07	-1.96	-2.87	-3.89	-4.94	-4.415	4
[19]	2.04	2.06	-1.93	-2.69	-3.98	-4.75	-4.365	7
[17]	3.11	2.66	-1.83	-1.31	-4.94	-3.98	-4.46	3
[143]	3.45	4.1	-0.53	-0.85	-3.98	-4.94	-4.46	2
[4]	2.05	2.04	-2.17	-1.87	-4.22	-3.91	-4.065	29
[82]	1.74	2.22	-2.42	-2.04	-4.16	-4.26	-4.21	26
[135]	3	2.79	-1.25	-1.44	-4.26	-4.22	-4.24	25
[119]	1.93	2.49	-1.84	-2.16	-3.77	-4.65	-4.21	19
[12]	2.04	2.04	-1.81	-1.98	-3.85	-4.02	-3.935	22
[30]	3.42	2.6	-1.22	-1.26	-4.65	-3.85	-4.25	18
[2]	3.72	3.98	-0.5	-0.67	-4.22	-4.65	-4.435	17
[a1]	4.94	4.22	0	0	-4.94	-4.22	-4.58	1

VIII. CONTROLLING ASPECT RATIO

Due to non-uniformity of number of cells per level, final aspect ratio is not rectangle. For better silicon utilization, it is required to make final aspect ratio as rectangle. Aspect ratio can be controlled while placing the cells by suitably modifying the algorithm “place-one-cell” given in Figure 14 which is discussed in the following paragraphs.

8.1 Algorithm: find-best-place

In the main algorithm, “place_circuit”, the following steps are added.

- Max_row=number of levels as given by synthesizer
- Total_width=total of widths of all cells in the circuit
- Average_width_per_level = Round (Total_width/Max_row) + Tolerance, where “Tolerance” is an integer to make the placement possible which can be varied based on need.

At the beginning, before starting placing cells, a layout surface rectangle of the size “Max_row X Average_width_per_level” is defined. As the placement progresses, the “used space” and “available space” are marked as shown in Figure 18.

The “find-best-place” algorithm works as follows.

- Current level of parent cell = c as shown in Figure 18.
- Check availability on level c-1.
- If space available on level c-1, place the leaf cell at level c-1.
- Else check availability on levels c, c-2, c+1 and c+2 in the “free” spaces as shown in Figure 18.
- Find the shortest free location from the current position shown as C in the Figure 18. Place the leaf cell here.

The example given in Figure 15 will be placed as follows.

The total number of levels excluding Primary Inputs and Primary Output are 4. Assuming that each cell has a width of unit 1, total width of all cells in the circuit is 15. So Max_row=4, Total_width=15 and Average_width_per_level = round (15/4) = 4. Here Tolerance = 0. So a maximum of 4 cells can be placed per row. The final placement by ANUPLACE is shown in Figure 19.

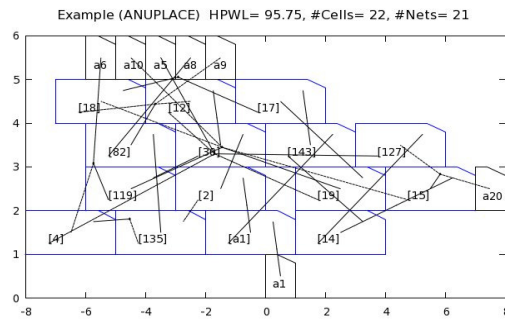


Figure 19 Final placement by ANUPLACE

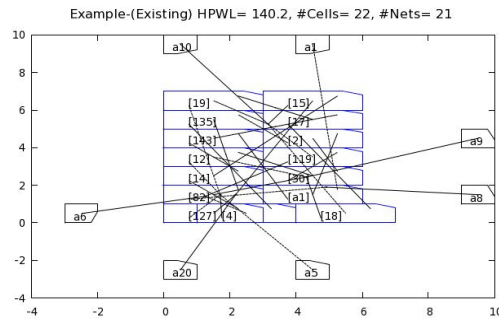


Figure 20 Placement by existing placer

The resulting layout is nearer to a rectangle. After placing the gates, Primary Inputs and Primary Outputs are placed nearer to the gates from which these Input/Outputs are taken. The placement given by the public domain placer [31,32,33,34] is shown in Figure 20.

The experimental set up to evaluate ANUPLACE using benchmark circuits and the results are given in the next section.

IX. RESULTS AND DISCUSSION

In this section, the test setup to evaluate the new algorithms with the benchmark circuits is described. The results are compared with those obtained from public domain placement algorithms. The test set up is shown in Figure 21. The test set up for comparing the results is shown in Figure 22. The benchmark circuit is taken in the form of a PLA. The normal placement bench mark circuits [36,37] are not useful, because they give only cell dimensions and interconnect information. Timing and other circuit information from synthesizer is not available in these placement bench marks. SIS synthesizer [35] is used for synthesizing the benchmark circuit. SIS [35] produces the net list in BLIF format along with the timing information. The BLIF output is then converted into Bookshelf format using the public domain tools available at the web site [31,32,33,34] using the utility "*blif2book-Linux.exe filename.blif filename*". ANUPLACE is used to place the circuit, which gives the placement output in Bookshelf format. To check the overlaps and also to calculate the wirelength (HPWL), a public domain utility [31,32,33,34], taken from the same web site, is used. The utility "*PlaceUtil-Lnx32.exe -f filename.aux -plotWNames filename -checkLegal -printOverlaps*", checks for out of core cells and overlaps. This utility also gives Half Perimeter Wire Length (HPWL) of the placed circuit. The same "BLIF" file is used with the public domain placer available at [31] using the utility "*LayoutGen-Lnx32.exe -f filename.aux -saveas filename*" and HPWL calculated using the utility "*PlaceUtil-Lnx32.exe -f filename.aux -plotWNames filename -checkLegal*".

The Table 4 shows the Half-Perimeter-Wire-Lengths (HPWL) of the placed circuits using existing public domain algorithms [31] and ANUPLACE. There is an average improvement of 53.7% in HPWLs with an average area penalty of 12.6%. Due to aligning of cells to signal flow, the layout produced by ANUPLACE is not a perfect rectangle. There will be white spaces at the left and right sides as shown in Figure 19. Because of this, there is an increase in the area of the placed circuit. The cells which are logically dependent are placed together as in [28]. Other placement algorithms randomly scatter the cells. Because of this there is reduction in HPWL of the entire placed circuit. Since the cells are placed along the signal flow, wire length along the critical paths will be optimum. So zigzags and criss-crosses are not present as in [24]. Circuit is naturally partitioned when trees are built rooted by Primary Outputs (POs). So there is no additional burden of extracting cones as in [23,28]. ANUPLACE is a constructive method, so better than other iterative methods. Only critical paths are given priority while construction of the layout. Global signal flow is kept in mind all through the placement, unlike other placement methods. Average slacks are used in these experiments. Using maximum of rise and fall slacks will give worst case delay. The timing results are being communicated in a separate paper. The final placement is closer to synthesis assumptions when compared to other placement methods. This approach may be useful towards evolving Synergistic

Design Flow, which is to create iteration loops that are tightly coupled at the various levels of design flow as mentioned in [1].

Table 4 Comparison of HPWL

Circuit Name	HPWL (ANUPLAC E)	HPWL (Existing)	Core cell Area	Area (Existing)	Area (ANUPLAC E)	Improvement in HPWL %	Area Penalty %
5xp1	1343.8	2021.9	317	361	378	50.46	4.7
9sym	4321.3	5162.4	657	729	730	19.46	0.1
apex2	10788.7	16088.1	1225	1369	1372	49.12	0.2
b12	765.1	1180.1	200	225	210	54.25	-6.7
b9	1591.1	2601.5	308	342	387	63.51	13.2
clip	2433	3968.9	511	576	612	63.13	6.3
cm82a	148.2	216	62	72	76	45.69	5.6
comp	2093.4	3681.9	452	506	650	75.88	28.5
con1	125.2	188.6	48	56	85	50.6	51.8
cordic	692.7	1569.8	230	256	360	126.63	40.6
count	2777.4	3842.9	473	529	520	38.36	-1.7
f51m	1494.7	2174.4	309	342	360	45.47	5.3
fa	62.9	83.9	30	36	44	33.27	22.2
ha	21.4	33.9	11	12	12	58.37	0
misex2	2107.3	2626.6	308	342	330	24.65	-3.5
mux1-8	111.3	148.3	32	36	42	33.33	16.7
mux8-1	130.6	211.9	59	64	88	62.29	37.5
o64	3008.3	3467.3	327	361	384	15.26	6.4
parity	346.3	636	149	169	196	83.64	16
rd53	425.2	659.5	130	144	192	55.09	33.3
rd73	1619.6	2666.1	387	420	500	64.61	19
rd84	1730.5	2588.6	394	441	480	49.59	8.8
sao2	1957.8	2913	384	420	500	48.79	19
squar5	648.9	835.2	156	169	192	28.71	13.6
t481	166.1	386.9	76	81	91	132.88	12.3
table3	69834.1	87642.5	4388	4900	4580	25.5	-6.5
Z5xp1	2521	3925.1	485	529	558	55.69	5.5
Z9sym	1276	1892.7	302	342	360	48.33	5.3

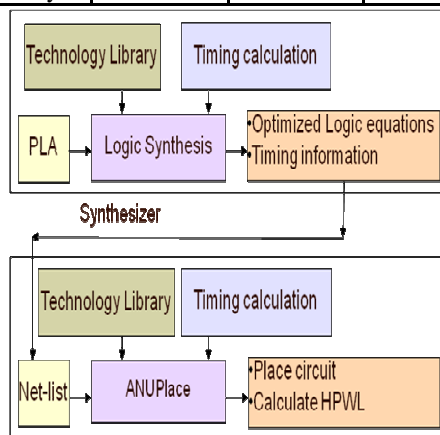


Figure 21 Test set up

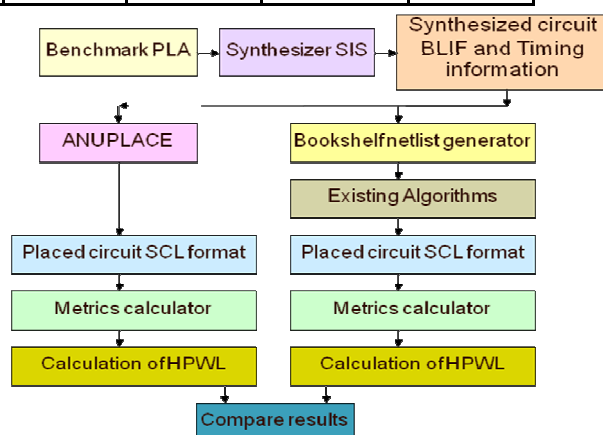


Figure 22 Set up to compare results

X. CONCLUSIONS AND FUTURE SCOPE

New algorithms place the circuits along signal flow as per the assumptions made during synthesis. The study conducted investigates the reasons for failure of placement tools to achieve timings given by the synthesizer. This showed us that certain assumptions made by synthesizer can be implemented, and some assumptions can never be implemented. Those which can be implemented are tried in our new placement algorithms. One problem encountered during implementation of the algorithms was that new placer produced cones, which are area inefficient. This problem to some extent circumvented by controlling the aspect ratio using non-critical cell placement to convert the cone into a rectangle. This new placer uses knowledge of the delay information during construction of the solution. This is useful to effectively control the aspect ratio of the placement solution. The improvements obtained in delay are being communicated in a separate paper.

ACKNOWLEDGEMENTS

We thank Dr. K.D. Nayak who permitted and guided this work to be carried out in ANURAG. We also thank members of ANURAG who reviewed the manuscript. Thanks are due to Mrs. D. Manikamma and Mr. D. Madhusudhan Reddy for the preparation of the manuscript.

REFERENCES

- [1] Kurt Keutzer., et al., (1997), "The future of logic synthesis and physical design in deep-submicron process geometries", *ISPD '97 Proceedings of the international symposium on Physical design*, ACM New York, NY, USA, pp 218-224.
- [2] Randal E. Bryant, et al., (2001), "Limitations and Challenges of Computer-Aided Design Technology for CMOS VLSI", *Proceedings of the IEEE*, Vol. 89, No. 3, pp 341-65.
- [3] Coudert, O, (2002), "Timing and design closure in physical design flows", *Proceedings. International Symposium on Quality Electronic Design (ISQED '02)*, pp 511 – 516.
- [4] Gosti, W., et al., (2001), "Addressing the Timing Closure Problem by Integrating Logic Optimization and Placement", *ICCAD 2001 Proceedings of the 2001 IEEE/ACM International Conference on Computer-aided design*, San Jose, California , pp 224-231.
- [5] Wilsin Gosti , et al., (1998), "Wireplanning in logic synthesis", *Proceedings of the IEEE/ACM international conference on Computer-aided design*, San Jose, California, USA, pp 26-33
- [6] Yifang Liu, et al., (2011), "Simultaneous Technology Mapping and Placement for Delay Minimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 30 No. 3, pp 416–426.
- [7] Pedram, M. & Bhat, N, (1991), "Layout driven technology mapping", *28th ACM/IEEE Design Automation Conference*, pp 99 – 105.
- [8] Salek, A.H., et al., (1999), "An Integrated Logical and Physical Design Flow for Deep Submicron Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 9, pp 1305–1315.
- [9] Naveed A. Sherwani, (1995), "Algorithms for VLSI Physical Design Automation", Kluwer Academic Publishers, Norwell, MA, USA.
- [10] Sarrafzadeh, M., & Wong, C.K., (1996), "An introduction to VLSI Physical Design", The McGraw-Hill Companies, New York.
- [11] Shahookar K & Mazumder P, (1991), "VLSI cell placement techniques" *ACM Computing Surveys*, Vol. 23, No. 2.
- [12] Jason Cong, et al., (2005), "Large scale Circuit Placement", *ACM Transactions on Design Automation of Electronic Systems*, Vol. 10, No. 2, pp 389-430.
- [13] Yih-Chih Chou & Young-Long Lin, (2001), "Performance-Driven Placement of Multi-Million-Gate Circuits", *ASICON 2001 Proceedings of 4th International Conference on ASIC*, Shanghai, China, pp 1-11.
- [14] Andrew B. Kahng & Qinke Wang, (2004), "An analytic placer for mixed-size placement and timing-driven placement", *Proceedings of International Conference on Computer Aided Design*, pp 565-572.
- [15] Jun Cheng Chi, et al., (2003), "A New Timing Driven Standard Cell Placement Algorithm", *Proceedings of International Symposium on VLSI Technology, Systems and Applications*, pp 184-187.
- [16] Swartz, W., & Sechen, C., (1995), "Timing Driven Placement for Large Standard Cell Circuits", *Proc. ACM/IEEE Design Automation Conference*, pp 211-215.

- [17] Tao Luo, et al., (2006), "A New LP Based Incremental Timing Driven Placement for High Performance Designs", *DAC '06 Proceedings of the 43rd Annual Design Automation Conference*, ACM New York, NY, USA, pp 1115-1120.
- [18] Carl Sechen & Alberto Sangiovanni-Vincentelli, (1985), "The TimberWolf Placement and Routing Package", *IEEE Journal of Solid-State Circuits*, vol. SC-20, No. 2, pp 510-522.
- [19] Wern-Jieh Sun & Carl Sechen, (1995), "Efficient and effective placement for very large circuits", *IEEE Transactions on CAD of Integrated Circuits and Systems*, Vol. 14 No. 3, pp 349-359
- [20] C. J. Alpert, et al., (1997), "Quadratic Placement Revisited", *34th ACM/IEEE Design Automation Conference*, Anaheim, pp 752-757
- [21] Rexford D. Newbould & Jo Dale Carothers, (2003), "Cluster growth revisited: fast, mixed-signal placement of blocks and gates", *Southwest Symposium on Mixed Signal Design*, pp 243 - 248
- [22] Andrew Kennings & Kristofer P. Vorwerk, (2006), "Force-Directed Methods for Generic Placement", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 25, NO. 10, pp 2076-2087.
- [23] Yu-Wen Tsay, et al., (1993), "A Cell Placement Procedure that Utilizes Circuit Structural Properties", *Proceedings of the European Conference on Design Automation*, pp 189-193.
- [24] Chanseok Hwang & Massoud Pedram, (2006), "Timing-Driven Placement Based on Monotone Cell Ordering Constraints", *Proceedings of the 2006 Conference on Asia South Pacific Design Automation: ASP-DAC 2006*, Yokohama, Japan, pp 201-206.
- [25] Brayton R K, et al., (1990), "Multilevel Logic Synthesis", *Proceedings of the IEEE*, Vol. 78, No. 2, pp-264-300.
- [26] Brayton R K, et al., (1987), "MIS: A Multiple-Level Logic Optimization System", *IEEE Transactions on Computer Aided Design*, Vol.6, No.6, pp-1062-1081.
- [27] Rajeev Murgai, et al., (1995), "Decomposition of logic functions for minimum transition activity", *EDTC '95 Proceedings of the European conference on Design and Test*, pp 404-410.
- [28] Cong, J. & Xu, D, (1995), "Exploiting signal flow and logic dependency in standard cell placement", *Proceedings of the Asian and South Pacific Design Automation Conference*, pp 399 – 404.
- [29] Fujita, M. & Murgai, R, (1997), "Delay estimation and optimization of logic circuits: a survey", *Proceedings of Asia and South Pacific Design Automation Conference*, Chiba, Japan, pp 25 – 30.
- [30] Andrew Caldwell, et al., (1999), "Generic Hypergraph Formats, rev. 1.1", from <http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Fundamental/HGraph/HGraph1.1.html>.
- [31] Saurabh Adya & Igor Markov, (2005), "Executable Placement Utilities" from <http://vlsicad.eecs.umich.edu/BK/PlaceUtils/bin>.
- [32] Saurabh N. Adya, et al., (2003), "Benchmarking For Large-scale Placement and Beyond", *International Symposium on Physical Design (ISPD)*, Monterey, CA, pp. 95-103.
- [33] Saurabh Adya and Igor Markov, (2003), "On Whitespace and Stability in Mixed-size Placement and Physical Synthesis", *International Conference on Computer Aided Design (ICCAD)*, San Jose, pp 311-318.
- [34] Saurabh Adya and Igor Markov, (2002), "Consistent Placement of Macro-Blocks using Floorplanning and Standard-Cell Placement", *International Symposium of Physical Design (ISPD)*, San Diego, pp.12-17.
- [35] Sentovich, E.M., et al., (1992), "SIS: A System for Sequential Circuit Synthesis", Memorandum No. UCB/ERL M92/41, Electronics Research Laboratory, University of California, Berkeley, CA 94720.
- [36] Jason Cong, et al, (2007), "UCLA Optimality Study Project", from <http://cadlab.cs.ucla.edu/~pubbench/>.
- [37] C. Chang, J. Cong, et al., (2004), "Optimality and Scalability Study of Existing Placement Algorithms", *IEEE Transactions on Computer-Aided Design*, Vol.23, No.4, pp.537 – 549.

AUTHORS

K. Santeppa obtained B.Tech. in Electronics and Communication engineering from JNTU and M Sc (Engg) in Computer Science and Automation (CSA) from Indian Institute of Science, Bangalore. He worked in Vikram Sarabhai Space Centre, Trivandrum from 1982 to 1988 in the field of microprocessor based real-time computer design. From 1988 onwards, he has been working in the field of VLSI design at ANURAG, Hyderabad. He received DRDO Technology Award in 1996, National Science Day Award in 2001 and "Scientist of the Year Award" in 2002. He is a Fellow of IETE and a Member of IMAPS and ASI. A patent has been granted to him for the invention of a floating point processor device for high speed floating point arithmetic operations in April 2002.



K.S.R. Krishna Prasad received B.Sc degree from Andhra University, DMIT in electronics from MIT, M.Tech. in Electronics and Instrumentation from Regional Engineering College, Warangal and PhD from Indian Institute of Technology, Bombay. He is currently working as Professor at Electronics and Communication Engineering Department, National Institute of Technology, Warangal. His research interests include analog and mixed signal IC design, biomedical signal processing and image processing.

